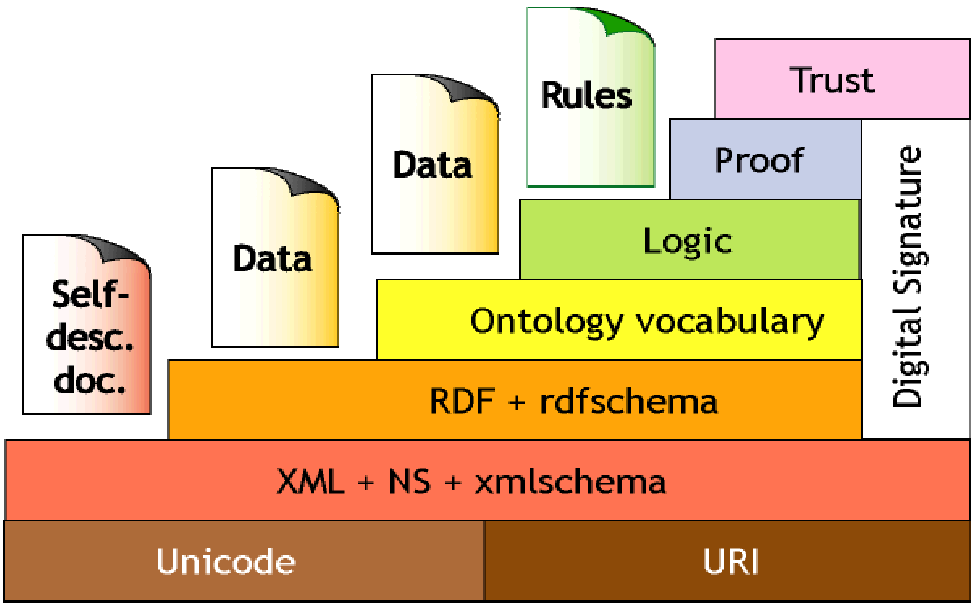
The Semantic Web

Lecture 3
The RDF layer

Riccardo Rosati

Dottorato in Ingegneria Informatica
Sapienza Università di Roma
a.a. 2006/07

The Semantic Web Tower



Syntax and semantics

- Syntax: the structure of data
- Semantics: the meaning of data
- Two conditions necessary for interoperability:
  - Adopt a **common syntax**: this enables applications to *parse* the data.
  - Adopt a **means for understanding** the semantics: this enables applications to *use* the data.

XML

- XML: eXtensible Mark-up Language
- XML documents are written through a user-defined set of tags
- tags are used to express the "semantics" of the various pieces of information

XML: example

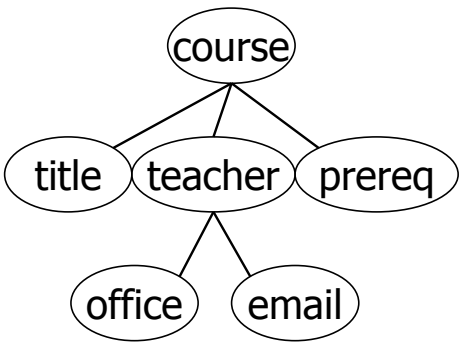
```
<course date="2007">
  <title>Seminari di Ingegneria del Software
  </title>
  <teacher>
    <name>Giuseppe De Giacomo</name>
    <email>degiacomo@dis.uniroma1.it</email>
  </teacher>
  <prereq>none</prereq>
</course>
```


XML

- XML: document = labelled tree
- node = label + attributes/values + contents

```
<course date="...">
  <title>...</title>
  <teacher>
    <office>...</office>
    <email>...</email>
  </teacher>
  <room>...</room>
  <prereq>...</prereq>
</course>
```

=



XML

- XML Schema = grammar for describing legal trees and datatypes
- can we use XML to represent semantics?

XML and semantics

```
<Predator>
      …
</Predator>
```

- Predator: a medium-altitude, long-endurance unmanned aerial vehicle system.
- Predator : one that victimizes, plunders, or destroys, especially for one's own gain.
- Predator : an organism that lives by preying on other organisms.
- ...

Limitations for semantic markup

- XML makes no commitment on:
  1. Domain-specific ontological vocabulary
  2. Ontological modeling primitives
- Requires pre-arranged agreement on 1 and 2
- Only feasible for closed collaboration:
  - agents in a small & stable community
  - pages on a small & stable intranet
- Not suited for sharing Web-resources

RDF

- RDF is a data model
  - the model is domain-neutral, application-neutral and ready for internationalization
  - the model can be viewed as directed, labeled graphs or as an object-oriented model (object/attribute/value)
- RDF data model is an abstract, conceptual layer independent of XML
  - consequently, XML is a transfer syntax for RDF, not a component of RDF
  - RDF data might never occur in XML form

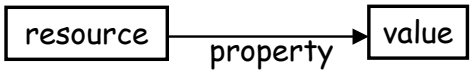
RDF model

RDF model = set of RDF triples

triple = expression (statement)

(subject, predicate, object)

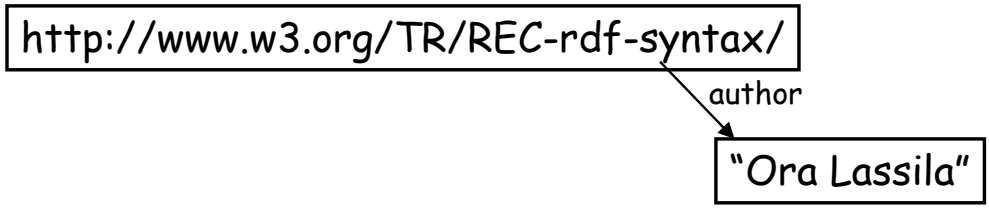
- subject = resource
- predicate = property (of the resource)
- object = value (of the property)



RDF triples

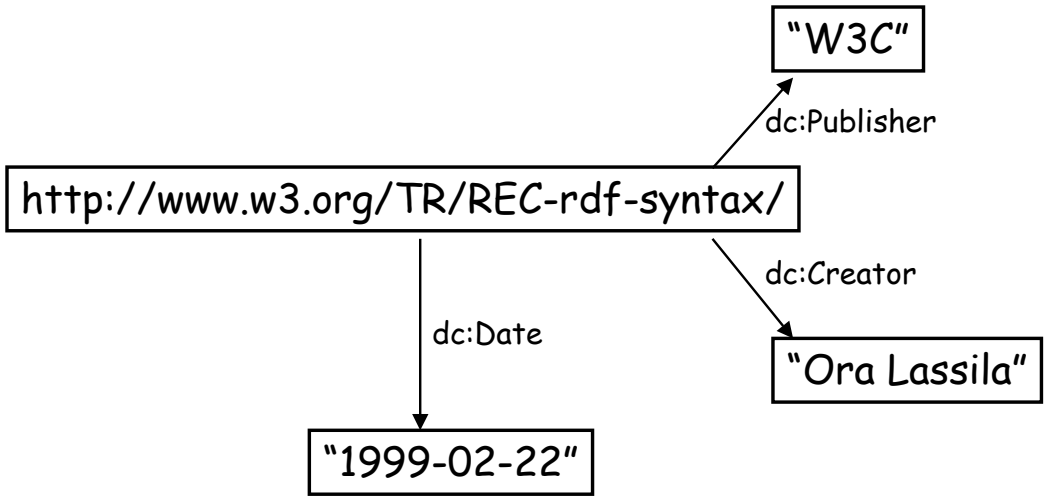
example: "the document at
http://www.w3c.org/TR/REC-rdf-syntax
has the author Ora Lassila"

triple:
http://www.w3c.org/TR/REC-rdf-syntax author "OraLassila"



⇒ RDF model = **graph**

RDF graph: example



Node and edge labels in RDF graphs

- node and edge labels:
  - URI
  - literal (string)
  - bnode (anonymous label)

but:

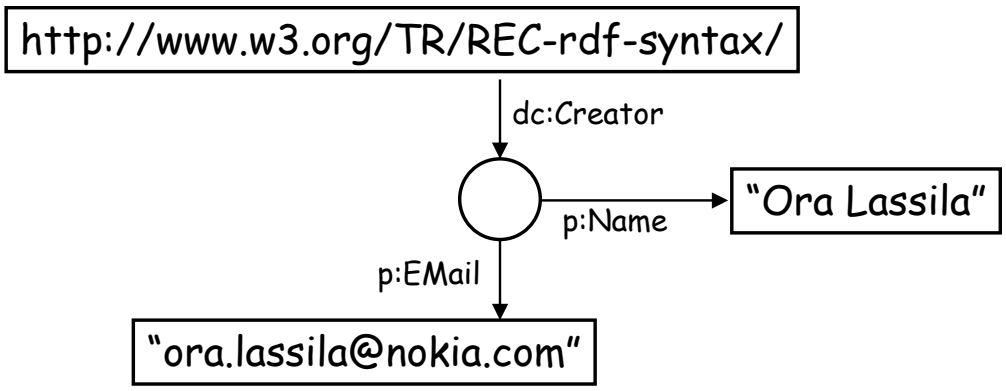
- a string cannot be the subject of a triple
- a graph node **can** be the label of an edge!
  (RDF has meta-modeling abilities)

Complex values

- values of properties need not be simple strings
- the value of a property can also be a graph node (corresponding to a resource)
  - arbitrarily complex tree and graph structures are possible
  - syntactically, values can be embedded (i.e., lexically in-line) or referenced (linked)

Complex values

example:



Blank nodes

- **blank node** (bnode) = RDF graph node with "anonymous label" (i.e., not associated with an URI)
- example:
- "John has a friend born the 21st of April"

```
ex:Jean   foaf:knows       _:p1
_:p1      foaf:birthDate   04-21
```

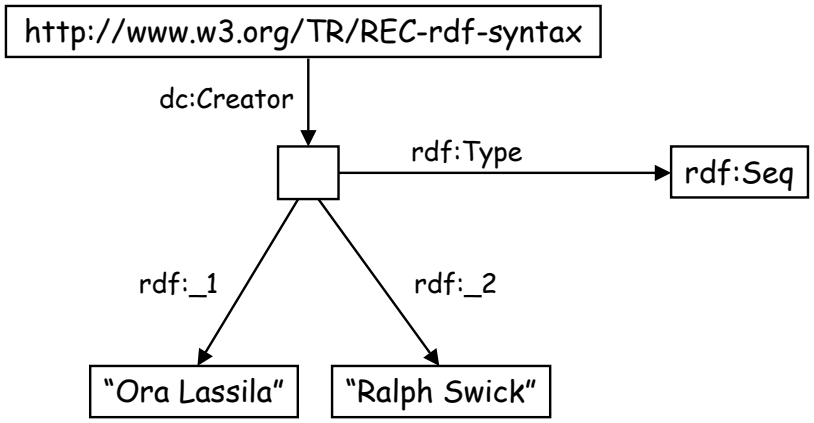
  - `_:p1` is the blank node (b-node)
- bnodes can be used both as subjects and objects

Containers

- Containers are collections
  - they allow grouping of resources (or literal values)
- It is possible to make statements about the container (as a whole) or about its members individually
- Different types of containers exist
  - bag - unordered collection
  - seq - ordered collection (= "sequence")
  - alt - represents alternatives
- It is also possible to create collections based on URI patterns
- Duplicate values are permitted (no mechanism to enforce unique value constraints)

Containers

example:



Higher-order statements

- One can make RDF statements about other RDF statements
  - example: "Ralph believes that the web contains one billion documents"
- Higher-order statements
  - allow us to express beliefs (and other modalities)
  - are important for trust models, digital signatures,etc.
  - also: metadata about metadata
  - are represented by modeling RDF in RDF itself

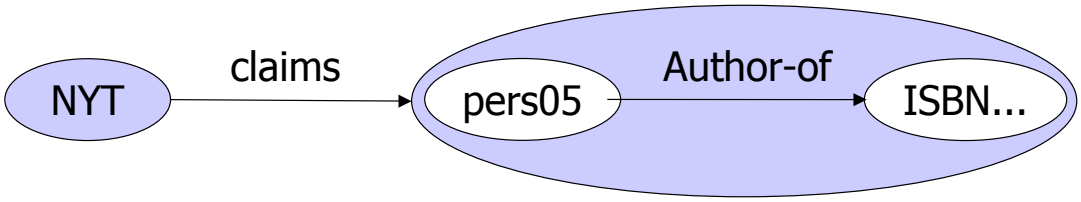
$\Rightarrow$ reification

Reification

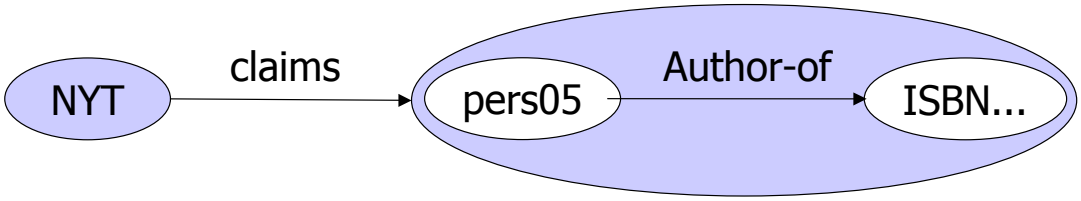
any RDF statement can be an object

⇒ reification

example:



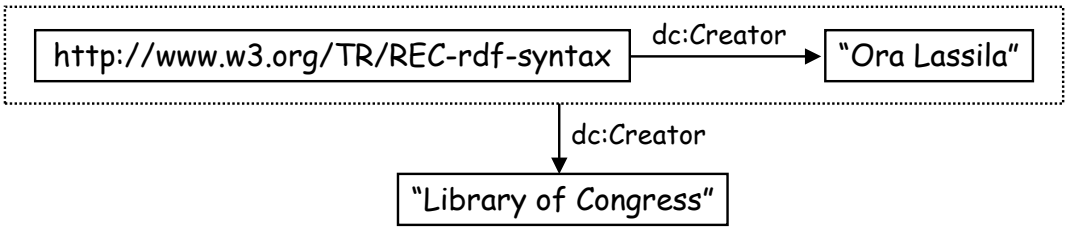
Reification




```
<rdf:Description rdf:about="#NYT">
  <claims>
    <rdf:Description rdf:about="#pers05">
      <authorOf>ISBN...</authorOf>
    </rdf:Description>
  </claims>
</rdf:Description>
```

Reification

- RDF provides a built-in predicate vocabulary for reification:



{ x, rdf:predicate, "dc:creator" }
{ x, rdf:subject, "http://www.w3.org/TR/RED-rdf-syntax }
{ x, rdf:object, "Ora Lassila" }
{ x, rdf:type, "rdf:statement" }

RDF syntaxes

RDF model = edge-labeled graph = set of triples

- graphical notation (graph)
- (informal) triple-based notation

  e.g., **`(subject, predicate, object)`**
- formal syntaxes:
  - N3 notation
  - Turtle notation
  - concrete (serialized) syntax: RDF/XML syntax

RDF syntaxes

- N3 notation:

```
subject predicate object .
```

- Turtle notation: example:

```
@prefix
    rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
:mary rdf:type <http://www.ex.org/Gardener>.
:mary :worksFor :ElJardinHaus.
:mary :name "Dalileh Jones"@en.
_:john :worksFor :ElJardinHas.
_:john :idNumber "54321"^^xsd:integer.
```

- concrete (serialized) syntax: RDF/XML syntax

RDF Schema

RDFS = RDF Schema

- Defines small **vocabulary** for RDF:
  - Class, subClassOf, type
  - Property, subPropertyOf
  - domain, range
- correspond to a set of RDF predicates:

  ⇒ meta-level

  ⇒special (predefined) “meaning”

RDFS

- vocabulary for defining **classes** and **properties**
- vocabulary for classes:
  - **`rdfs:Class`** (a resource is a class)
  - **`rdf:type`** (a resource is an instance of a class)
  - **`rdfs:subClassOf`** (a resource is a subclass of another resource)

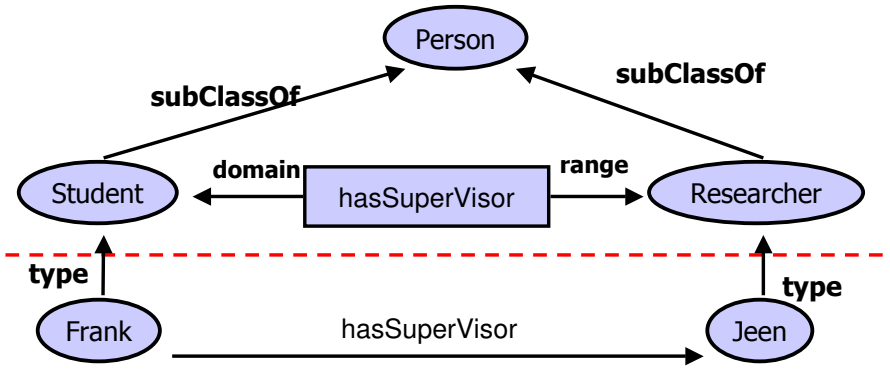
RDFS

vocabulary for properties:

- **`rdfs:Property`** (a resource is a property)
- **`rdfs:domain`** (denotes the first component of a property)
- **`rdfs:range`** (denotes the second component of a property)
- **`rdfs:subPropertyOf`** (expresses ISA between properties)

RDFS

example:



RDFS

example (classes):

```
(ex:MotorVehicle, rdf:type, rdfs:Class)
(ex:PassengerVehicle, rdf:type, rdfs:Class)
(ex:Van, rdf:type, rdfs:Class)
(ex:Truck, rdf:type, rdfs:Class)
(ex:MiniVan, rdf:type, rdfs:Class)
(ex:PassengerVehicle, rdfs:subClassOf,
  ex:MotorVehicle)
(ex:Van, rdfs:subClassOf,  ex:MotorVehicle)
(ex:Truck, rdfs:subClassOf,  ex:MotorVehicle)
(ex:MiniVan, rdfs:subClassOf, ex:Van)
(ex:MiniVan, rdfs:subClassOf, ex:PassengerVehicle)
```

RDFS

example (classes):

RDFS

example (properties):

```
(ex:weight, rdf:type, rdfs:Property)
(ex:weight, rdfs:domain, ex:MotorVehicle)
(ex:weight, rdfs:range, Integer)
```

RDFS: meta-modeling abilities

example (meta-classes):

```
(ex:MotorVehicle, rdf:type, rdfs:Class)
(ex:myClasses, rdf:type, rdfs:Class)
(ex:MotorVehicle, rdf:type,  ex:myClasses)
```


RDFS: XML syntax

example:

```
<rdf:Description ID="MotorVehicle">
   <rdf:type resource="http://www.w3.org/...#Class"/>
   <rdfs:subClassOf
   rdf:resource="http://www.w3.org/...#Resource"/>
</rdf:Description>

<rdf:Description ID="Truck">
   <rdf:type resource="http://www.w3.org/...#Class"/>
   <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
</rdf:Description>
```

RDFS: XML syntax

example (cont.):

```
<rdf:Description ID="registeredTo">
   <rdf:type resource="http://www.w3.org/...#Property"/>
   <rdfs:domain rdf:resource="#MotorVehicle"/>
   <rdfs:range rdf:resource="#Person"/>
</rdf:Description>

<rdf:Description ID="ownedBy">
   <rdf:type resource="http://www.w3.org/...#Property"/>
   <rdfs:subPropertyOf rdf:resource="#registeredTo"/>
</rdf:Description>
```


RDF + RDFS: semantics

- what is the exact meaning of an RDF(S) graph?
- initially, a formal semantics was not defined!
- main problems:
  - bnodes
  - meta-modeling
  - formal semantics for RDFS vocabulary
- recently, a model-theoretic semantics has been provided

⇒formal definition of entailment and query answering over RDF(S) graphs

Incomplete information in RDF graphs

- bnodes = existential values (null values)

$\Rightarrow$ introduce incomplete information in RDF graphs

- an RDF graph can be seen as an incomplete database represented in the form of a naive table
- an RDF graph is represented by a unique table T, with values being constants or named existential variables
- an RDF graph can be seen as a boolean conjunctive query over the unique relation T

RDF + RDFS: semantics

problems with meta-data:

```
(#a rdf:type #C)
(#C rdf:type #R)
(#R rdf:type #a)
```

or

```
(#C rdf:type #C)
```

are correct (formally meaningful) RDF statements

$\Rightarrow$ but no intuitive semantics

Formal semantics for RDF + RDFS

- formal semantics for RDFS vocabulary
- RDFS statements = **constraints** over the RDF graph
- entailment in RDF + RDFS = reasoning (query answering) over an **incomplete database with constraints**

Querying RDF: SPARQL

recursive acronym:

SPARQL Protocol and RDF Query Language

- W3C standardisation effort similar to the XQuery query language for XML data
- Data Access Working Group (DAWG)
- suitable for remote use (remote access protocol)

SPARQL

example:

```
PREFIX
   abc: <http://mynamespace.com/exampleOntology#>
SELECT ?capital ?country
WHERE { ?x abc:cityname ?capital.
        ?y abc:countryname ?country.
        ?x abc:isCapitalOf ?y.
        ?y abc:isInContinent abc:africa. }
```


SPARQL

```
PREFIX
   abc: <http://mynamespace.com/exampleOntology#>
SELECT ?capital ?country
WHERE { ?x abc:cityname ?capital.
        ?y abc:countryname ?country.
        ?x abc:isCapitalOf ?y.
        ?y abc:isInContinent abc:africa. }
```

- Variables are outlined through the "?" prefix ("$" is also possible).
- The ?capital and the ?country will be returned.
- The SPARQL query processor returns all hits matching the pattern of the four RDF-triples.
- "property orientation" (class matches can be conducted solely through class-attributes/properties)

SPARQL

- **basic graph pattern** (BGP) query = RDF graph with possibly variables as labels
- SPARQL defines on top of a BGP additional operators (AND, FILTER, UNION, OPTIONAL)

RDF in the real world

- RDF and SPARQL are W3C standards
- Widespread use for metadata representation, e.g.
  - Apple (MCF)
  - Adobe (XMP)
  - Mozilla/Firefox
- Oracle supports RDF, and provides an extension of SQL to query RDF data
- HP has a big lab (in Bristol) developing specialized data stores for RDF (Jena)
- ...but: research is beyond practice

Coming next: RDFS vs. OWL